이진 자켓 비트열의 VLSI 구조

정회원 박 주 용*, 이 문 호**

A VLSI Architecture for the Binary Jacket Sequence

Ju-yong Park*, Moon-ho Lee** *Regular Members*

요 약

자켓 행렬(Jacket matrix)은 왈쉬 하다마드(Walsh Hadamard) 행렬 구조를 바탕으로 확장한 행렬이다. 왈쉬 하다마드 행렬이 +1, -1을 기본 원소로 하고 있는 반면 자켓 행렬은 ±1과 $\pm\omega(\pm j, \pm 2^n)$를 각각 원소로 가질 수 있다. 이 행렬은 중앙 부근에 무게(weight)를 갖는데, 하다마드 행렬 크기의 1/4 크기로 부호 부분과 무게 부분으로 구성된다.

본 논문에서는 기존에 행렬 중앙에 강제적으로 무게를 할당하여 자켓 행렬을 구성하였으나, 어떠한 크기의 행렬도 크기와 무게만 정해주면 생성해낼 수 있는 이진 인덱스를 이용한 간단한 비트열 형태의 일반식이 제시된다. 무게는 행과 열의 이진 인덱스의 최상위 두 비트를 Exclusive-OR 연산한 결과가 1인 원소에 부여된다. 또한 분산연산(Distributed Arithmetic:DA) 알고리즘을 이용한 고속자켓변환(Fast Jacket Transform)의 VLSI 구조를 제시한다. 자켓 행렬은 cyclic한 특성을 가지고 있어서 암호화, 정보 이론 및 WCDMA의 복소수 확산 QPSK 변조부에 응용될 수 있다.

ABSTRACT

The Jacket matrix is based on the Walsh-Hadamard matrix and an extension of it. While elements of the Walsh-Hadamard matrix are +1 or -1, those of the Jacket matrix are ±1 and ±ω, which is ±j and $\pm 2^n$. This matrix has weights in the center part of the matrix and its size is 1/4 of Hadamard matrix, and it has also two parts, sign and weight.

In this paper, instead of the conventional Jacket matrix where the weight is imposed by force, a simple Jacket sequence generation method is proposed. The Jacket sequence is generated by AND and Exclusive-OR operations between the binary indices bits of row and those of column. The weight is imposed on the element only when the product of each Exclusive-OR operations of significant upper two binary index bits of a row and column is 1. Each part of the Jacket matrix can be represented by Jacket sequence using row and column binary index bits. Using Distributed Arithmetic (DA), we present a VLSI architecture of the Fast Jacket transform is presented. The Jacket matrix is able to be applied to cryptography, the informationtheory and complex spreading Jacket QPSK modulation for WCDMA.

I. 서 론

하다마드(Hadamard) 행렬(matrix)은 직교성(orthogonality)을 가지며, 음성신호와 영상신호의 변환 및 부호화에 매우 유용하게 쓰인다[1]. 하다마드 변환은 하다마드 행렬 원소 중 +1, -1에 의해 이루어지므로, 단지 신호의 가산과 감산만으로도 변환을 수행할 수 있다. 최근에 하다마드 행렬은 영상 부호화 분야, CDMA 대역확산통신이나 암호화에서 데이터 비트열을 표현하는데 사용되고 있다.

* 서남대학교 전기전자멀티미디어공학부(jypark@tiger.seonam.ac.kr)
** 전북대학교 전자정보공학부, 정보통신연구소(moonho@chonbuk.ac.kr)
논문번호 : 00147-0505, 접수일자 : 2000년 5월 5일

리버스 자켓(Reverse Jacket) 행렬은 하다마드 행렬을 사용하여 하중 하다마드(WH : Weighted Hadamard) 행렬로부터 구성되었다[10]. 일반적인 WH와 하다마드 행렬을 포함하여 리버스 자켓을 간단히 자켓 행렬(Jacketmatrix)이라 한다[1],[2],[4],[10]. 자켓 행렬은 하다마드 행렬의 중심부에 무게(weight) $\omega(\pm jn, \pm 2^n, n=1,2,3,\cdots)$를 가지며, 무게가 있는 부분과 없는 부분이 순방향과 역방향 치환에 의해 무게의 위치만 변환되는 순환적인 구조로 되어 있고, 이때, 부호는 변하지 않는다. 기존의 리버스 자켓 행렬은 행렬 연산법을 기본으로 하여 이루어져서[2], 오류정정, 암호화 및 QPSK 변조에 사용하기에 어려움이 있었다. 따라서 $2^n \times 2^n$ 크기의 자켓 행렬을 쉽게 구현하고 부호화에 이용하기 위해서는 비트열로 표현되는 일반식이 필요하였으나 일반식을 발견하지 못했었다. 그러나 이 일반식은 행과 열의 이진 인덱스를 Exclusive-OR 와 AND 연산으로 나타낼 수 있음이 발견되었다[4]. 이러한 방법은 기존의 자켓 행렬 생성법 보다 간단할 뿐만 아니라 구현면에서도 효율적이다. 자켓 부호는 자켓 비트열(Jacket Sequence)로 부터 생성할 수 있고 실수 영역에서 확장하여 복소수 영역에서도 다중레벨(multi-level)로 나타낼 수 있다[15].

DSP(Digital Signal Processing)응용의 여러 분야에서 신호 변수는 푸리에(Fourier) 전력 스펙트럼을 사용하여 측정된다. 푸리에 변환 계산은 비교적 복잡하고 하드웨어 효율면이나 변수 측정의 정확성면에서 중요하게 사용된다[18]. 이러한 분야에 고속 하다마드 변환(Fast Hadamard Transform : FHT)이 사용되며 이는 적응형 필터나 DFT(Discrete Fourier Transform) 스펙트럼 필터구현을 위한 DFT 계산에도 효율적이라고 알려져 있다.

일반적인 주파수 영역에서의 FIR필터는 왈쉬 주파수 영역으로 쉽게 변환될 수 있고 유사하게 유한 충격파(impulse) 응답 필터 구현에도 이러한 결과를 이용할 수 있다. 1-D FHT를 기반으로 2-D FHT를 구현하게 되면 손실 없는 이미지 압축이 가능하게 된다[18]. FHT는 직교 변환 이므로 단지 가산과 감산만으로 구현이 가능하고 sinusoidal 같은 변환보다 더 빠른 성능을 보이며 DFT와 유사하게 행렬-벡터 곱으로 나타낼 수 있다[19]. 즉, N 입력일 경우 $Nlog2N$ 가산과 감산으로 고속 알고리즘을 가지게 된다[5],[20].

FHT는 DFT와 유사하고 순환적 구조(cyclic structure)를 가진다. 하다마드 변환의 시스톨릭(Systolic)한 배열형태의 칩은 South California University에서 설계되었고 MOSIS가 제작하였다[20]. 이 칩은 $2N$-1 클럭 사이클과 Latency N이 필요하지만 배열에서 가산 구현은 word level 이므로 시간은 $log2N+n$에 비례하여 증가한다. n은 입력 샘플의 워드 길이이다. 새롭게 제안된 비트 레벨 시스톨릭 구조의 속도를 증가시키기 위한 기법이 [19]에서 제안되었다. 이것은 latency가 $N(n+log2N)$이고, $(2N-1)\cdot(n+log2N)$의 클럭 사이클을 필요로 한다.

ROM에 근거한 분산연산(DA : Distributed Arithmetic)[20]라 불리는 기존의 DA는 미리 계산된 데이터를 발생시키기 위하여 비트레벨(bit level)의 내적으로 다양한 입력을 분해 시킨다. ROM에 근거한 DA는 미리 계산된 데이터를 롬에 저장하고 그것을 규칙적인 형태로 이용함으로써 VLSI 구현 시 면적면에서 효율적이다. 하지만 inner product의 크기가 증가할 때 ROM 면적은 지수적으로 증가하게 되고 면적 효율이 떨어지게 된다[21]. DA based ROM를 사용하게 되면 효율적인 구현이 가능하게 된다. 기본적으로 여러 ROM 비트 열과 가산, 감산, 입력 데이터의 천이(shift) 연산이 필요하다. 이러한 모든 함수들이 FPGA에 효율적으로 대응(mapping)된다[22].

자켓 행렬은 cocyclic한 특성 때문에 암호화의 벤트함수(Bent function)[12], 정보 이론 및 WCDMA의 복소 스프레딩 QPSK 변조부등 더욱 다양한 응용분야를 가질 수 있고[13], Space Time Code에서 대역효율, 전력면에서도 효율적인 특성을 나타낸다.

본 논문에서는 자켓 행렬을 새롭게 비트열 형태로 표현할 수 있도록 행과 열의 인덱스를 이진수로 변환한 후 이 이진 비트들을 이용하여 일반식을 제안하고, 고속 자켓 변환(fast Jacket transform:FJT) 알고리즘을 하드웨어로 설계하기 위한 VLSI 구조를 보인다. 2장에서는 자켓 행렬을 소개하고, 3장에서는 자켓 비트열의 이진 인덱스를, 4장에서는 1차원 FJT의 하드웨어 구현 방법 을 보이고 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 자켓 행렬

Central Limit 이론과 가우시안 함수처럼, 자켓 행렬도 공간 영역에서 행렬의 중심부에 무게를 가진다. 하다마드 함수는 시간 영역에서 단지 두 레벨(+1 and -1)로 신호파형이 표현된다. 그러나, 두 레벨만으로는 신호를 정확히 표현할 수 없으므로[1],

연속적인신호를 표현하기 위해서는 Haar함수, Slant 함수 같은 다중레벨 함수가 더 효과적이다. 하다마드 함수의 중심부에 무게를 둔 자켓 함수는 멀티레벨로 표현되기 때문에 신호를 보다 정확히 나타낼 수 있다. 그림 1은 ω=2인 시간영역에서의 멀티레벨 연속 자켓 함수를 보여주고 있으며 점선 부분은 무게가 가해진 부분을 표시 한다. 이 무게에 따라서 비트열의 레벨이 다양하게 표시 될 수 있다.

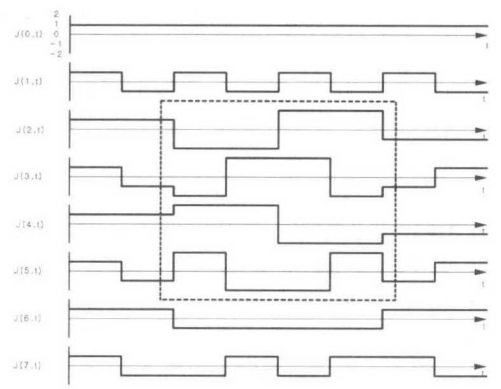


그림 1. 시간영역에서의 멀티레벨 연속 자켓 함수, ω=2.

$N\times N$, $(N=2^n)$자켓 행렬을 $[J]_n$이라 하고, $n\geq 1$ 이며, 무게 ω는 $\pm j$와 $\pm 2^k$이고 k는 정수이다. 이때 무게는 행렬의 중심부에 위치하며, $[J]_2$와 그의 역변환 $[J]_2^{-1}$는 다음과 같다.

$$[J]_2=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -\omega & \omega & -1\\ 1 & \omega & -\omega & -1\\ 1 & -1 & -1 & 1\end{bmatrix}, \tag{1}$$

$$[J]_2^{-1}=\frac{1}{4\omega}\begin{bmatrix}\omega & \omega & \omega & \omega\\ \omega & -1 & 1 & -\omega\\ \omega & 1 & -1 & -\omega\\ \omega & -\omega & -\omega & \omega\end{bmatrix}, \tag{2}$$

단, $[J]_2[J]_2^{-1}=[I]_2$,$[I]_2$는 4×4 항등행렬이다.

$[J]_n$은 무게에 따라 다양한 분야에 응용될 수 있다. 표 1은 다양한 무게를 가진 자켓 행렬을 보여주는데, 만약 =1이면 자켓 행렬은 하다마드 행렬과 같고 ω=2 이면 하중 하다마드 행렬이된다. $\omega=j$인 경우는 복소수 영역에서 직교성을 가지는 독특한 행렬이 된다[8]. 식 (1)과 식 (2)에서 볼 수 있듯이 순방향과 역방향 행렬은 뒤집을 수 있는 자켓처럼 무게 부분이 서로 교환되는 흥미 있는 구조를 가진다. ω=1, j 인 경우, 자켓 행렬은 직교성을 가지며, 특히, $\omega=j$ 인 경우는 WCDMA의 복소 확산 QPSK 변조에서 응용될 수 있는 이점을 가진다[13].

표 1. 여러 무게에 따른 자켓 행렬의 예, ω =1, 2, j.

weight \ Jacket matrix	$[J]_2$	$[J]_2^{-1}$
$\omega=1$	$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}$	$\frac{1}{4}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}$
$\omega=2$	$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -2 & 2 & -1\\ 1 & 2 & -2 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}$	$\frac{1}{8}\begin{bmatrix}2 & 2 & 2 & 2\\ 2 & -1 & 1 & -2\\ 2 & 1 & -1 & -2\\ 2 & -2 & -2 & 2\end{bmatrix}$
$\omega=j$	$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -j & j & -1\\ 1 & j & -j & -1\\ 1 & -1 & -1 & 1\end{bmatrix}$	$\frac{1}{4j}\begin{bmatrix}j & j & j & j\\ j & -1 & 1 & -j\\ j & 1 & -1 & -j\\ j & -j & -j & j\end{bmatrix}$

표 1은 우리가 뒤집어 입을 수 있는 자켓 옷의 형태와 유사한 순방향과 역방향 자켓 행렬이다.

자켓 행렬은 순방향과 역방향 행렬 모두 무게가 있는 부분과 없는 부분이 치환에 의해 부호는 변함없이 무게의 위치만 서로 바뀌는 흥미 있는 순환 구조로 되어있다. 하다마드 행렬처럼 이러한 구조는 더 높은 차수의 행렬을 만들어 낼 수 있게 한다.

$$[J]_{n+1}=[J]_n\otimes[H]_1 \tag{3}$$

여기서 $n\geq 2$, $\otimes$는 크로네커 곱(kronecker product) 이고, $[H]_1$은 하다마드 기본 행렬이다.

$$[H]_1=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}. \tag{4}$$

예를 들어, 식 (1), 식 (2), 식 (3), 식 (4)로부터 n이 3 이고 무게가 ω인 행렬의 순방향과 역방향 형태를 나타내면 다음과 같다.

$$[J]=\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1\\ 1 & 1 & -\omega & -\omega & \omega & \omega & -1 & -1\\ 1 & -1 & -\omega & \omega & \omega & -\omega & -1 & 1\\ 1 & 1 & \omega & \omega & -\omega & -\omega & -1 & -1\\ 1 & -1 & \omega & -\omega & -\omega & \omega & -1 & 1\\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1\\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1\end{bmatrix},\quad [J]^{-1}=\frac{1}{2\omega}\begin{bmatrix}\omega & \omega & \omega & \omega & \omega & \omega & \omega & \omega\\ \omega & -\omega & \omega & -\omega & \omega & -\omega & \omega & -\omega\\ \omega & \omega & -1 & -1 & 1 & 1 & -\omega & -\omega\\ \omega & -\omega & -1 & 1 & 1 & -1 & -\omega & \omega\\ \omega & \omega & 1 & 1 & -1 & -1 & -\omega & -\omega\\ \omega & -\omega & 1 & -1 & -1 & 1 & -\omega & \omega\\ \omega & \omega & -\omega & -\omega & -\omega & -\omega & \omega & \omega\\ \omega & -\omega & -\omega & \omega & -\omega & \omega & \omega & -\omega\end{bmatrix} \tag{5}$$

$(2^n\times 1)$입력 데이터 행렬인 $[X]_n$이 주어진 경우

2^n, 1차원 자켓 행렬의 변환 출력 $[Y]_n$은

$$[Y]_n(i)=\sum_{j=0}^{2^n-1}[J]_n(i,j)[X]_n(j) \quad (6)$$

이다. 여기서 i =0,1,2,…, 2^n-1이다. 식 (6)의 역 변환은 다음과 같다.

$$[X]_n(j)=\sum_{i=0}^{2^n-1}[Y]_n(i)[J]_n^{-1}(i,j) \quad (7)$$

여기서 j =0,1,2,…, 2^n-1이다.

Ⅲ. 자켓 비트열의 이진 인덱스

그림 1에서 보여준 자켓 함수의 샘플링에 의해 간단한 불연속 자켓 비트열을 만들 수 있다. 그 예로, 표 2에서 8×8 자켓 행렬 $[J]_3$ 을 보이고 있는데, 표 3에서 보듯이 행과 열의 인덱스 i, j를 이진수 형태인 $i_2i_1i_0$와 $j_2j_1j_0$로 각각 나타낼 수 있다. 무게는 행렬의 크기에 관계없이 항상 행렬의 중앙에 위치한다. 행렬은 부호 부분과 무게 부분으로 나누어지고 두 부분 모두 이진 인덱스의 조합으로 표현할 수 있다.

표 3에서는 $[J]_3$을 예로 들어 카르노 맵을 이용한 무게 결정방법을 보여준다. 표 3에서 알 수 있듯이 무게계수를 결정할 때는 인덱스의 최상위 두 비트만이 이용되며, 이때 무게는 $(\omega)^{W_n(i,j)}$와 같다.

표 2. 자켓 행렬 $[J]_n$의 무게계수, n=3, ω=2.

index i \ index j	$j_2j_1j_0$	0	1	2	3	4	5	6	7
$i_2i_1i_0$		000	001	010	011	100	101	110	111
0	000	1	1	1	1	1	1	1	1
1	001	1	-1	1	-1	1	-1	1	-1
2	010	1	1	-2	-2	2	2	-1	-1
3	011	1	-1	-2	2	2	-2	-1	1
4	100	1	1	2	2	-2	-2	-1	-1
5	101	1	-1	2	-2	-2	2	-1	1
6	110	1	1	-1	-1	-1	-1	1	1
7	111	1	-1	-1	1	-1	1	1	-1

$$W_n(i,j)=(i_{n-1}\oplus i_{n-2})(j_{n-1}\oplus j_{n-2}) \quad (8)$$

여기서, $i=(i_{n-1}i_{n-2}i_1i_0)_2\cdots$, $j=(j_{n-1}j_{n-2}\cdots j_1j_0)_2$, $i_k,j_k=\begin{cases}0 or 1 & \text{for } 0\le k\le n-1\\ 0 & \text{for } otherwise\end{cases}$ 이고 $i,j=0,1,\cdots,2^{n-1}$, $n\ge 1$이다.

이때 무게를 가지는 원소의 수는 총 행렬 원소의 1/4(=1/2×1/2)이고, 행과 열의 이진 인덱스 조합으로 카르노 맵에 의해 W_n의 대수 표현식을 유도해 낼 수 있다.

예 1 : n=3인 경우, $[J]_3$의 무게 계수를 식 (8)로부터 구해보면 다음과 같다.

$$W_3(i,j)=i_2\overline{i_1}j_2j_1+\overline{i_2}i_1\overline{j_2}j_1+\overline{i_2}i_1j_2\overline{j_1}+i_2\overline{i_1}j_2\overline{j_1}$$
$$=(i_2\oplus i_1)(j_2\oplus j_1) \quad (9)$$

단, $i,j=0,1,\cdots,7$, $i=i_2i_1i_0$, $j=j_2j_1j_0$ 이다.

행렬 원소의 부호 역시 표 2와 4에 의해 결정 된다. 표 4는 $[J]_3$부호 결정을 위한 카르노맵을 나타내었다. 행렬 원소의 부호계수 S_n은 각 행과 열의 이진 인덱스의 조합에 의해 결정되고 이때 부호는 $(-1)^{S_n(i,j)}$가 된다.

표 3. $[J]_3$의 무게 결정을 위한 카르노맵.

$i_2i_1i_0$ \ $j_2j_1j_0$	000	001	011	010	110	111	101	100
000								
001								
011			1	1			1	1
010			1	1			1	1
110								
111								
101			1	1			1	1
100			1	1			1	1

$$S_n(i,j)=\sum_{k=0}^{n-1}\oplus i_kj_k$$
$$=i_{n-1}J_{n-1}\oplus i_{n-2}j_{n-2}\cdots i_1j_1\oplus i_0j_0 \quad (10)$$

여기서, $i=(i_{n-1}i_{n-2}\cdots i_1i_0)_2$, $j=(j_{n-1}j_{n-2}\cdots j_1j_0)_2$, $i_k,j_k=\begin{cases}0 or 1 & \text{for } 0\le k\le n-1\\ 0 & \text{for } otherwise\end{cases}$ 이고 $i,j=0,1,\cdots,2^{n-1}$, $n\ge 1$이다.

예 2 : 예 1과 유사한 방법으로, n=3인 경우, $[J]_3$의 부호 계수 결정은 다음과 같다.

$$S_3(i,j)=\sum_{k=0}^{2}\oplus i_kj_k \quad (11)$$

단, $i,j=0,1,\cdots,7$, $i=i_2i_1i_0$, $j=j_2j_1j_0$ 이다.

표 4. $[J]_3$의 부호 결정을 위한 카르노맵.

$i_2i_1i_0$ \ $j_2j_1j_0$	000	001	011	010	110	111	101	100
000								
001		1	1			1	1	
011		1		1	1		1	
010			1	1	1	1		
110			1	1			1	1
111		1		1		1		1
101		1	1		1			1
100				1	1	1	1	

자켓 행렬의 모든 원소에 관한 부호 $(-1)^{S_3(i,j)}$는 ω=2인 경우에 대해 표 2에 나타냈다. 행렬 크기의 확장을 위해 자켓 비트열의 일반식을 무게요소와 부호의 곱으로 다음 식과 같이 나타낼 수 있다.

$$[J]_n(i,j)=(-1)^{S_n}(i,j)(\omega)^{W_n}(i,j) \tag{12}$$

단, $i=(i_{n-1}i_{n-2}\cdots i_1i_0)_2$, $j=(j_{n-1}j_{n-2}\cdots j_1j_0)_2$, $i_k,j_k=\begin{cases}0 or 1 & \text{for } 0\le k\le n-1\\ 0 & \text{for } otherwise\end{cases}$이고 $i,j=0,1,\cdots,2^{n-1}$, $S_n(i,j)=\sum_{k=0}^{n-1}\oplus i_kj_k$ $W_n(i,j)=(i_{n-1}\oplus i_{n-2})(j_{n-1}\oplus j_{n-2})$, $n\ge1$이다.

자켓 행렬의 역행렬, $[J]_n^{-1}$은 식 (12)에서 ω를 ω^{-1}로 바꾸고 $\frac{1}{2^n}$로 스케일링하여 식 (13)과 같이 나타낼 수 있다.

$$[J]_n^{-1}(i,j)=\frac{1}{2^n}(-1)^{S_n(i,j)}(\omega^{-1})^{W_n(i,j)}. \tag{13}$$

예3 : 식 (11)과 식 (13)으로부터 n=3, ω=j인 경우 $[J]_n$와 $[J]_n^{-1}$를 나타내면 다음과 같다.

$$[J]_3=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&-1&1&-1&1&-1&1&-1\\1&1&-j&-j&j&j&-1&-1\\1&-1&-j&j&j&-j&-1&1\\1&1&j&j&-j&-j&-1&-1\\1&-1&j&-j&-j&j&-1&1\\1&1&-1&-1&-1&-1&1&1\\1&-1&-1&1&-1&1&1&-1\end{bmatrix},\quad [J]_3^{-1}=\frac{1}{8}[J]_3^{\dagger}=\frac{1}{8}\begin{bmatrix}1&1&1&1&1&1&1&1\\1&-1&1&-1&1&-1&1&-1\\1&1&j&j&-j&-j&-1&-1\\1&-1&j&-j&-j&j&-1&1\\1&1&-j&-j&j&j&-1&-1\\1&-1&-j&j&j&-j&-1&1\\1&1&-1&-1&-1&-1&1&1\\1&-1&-1&1&-1&1&1&-1\end{bmatrix} \tag{14}$$

Ⅳ. 1차원 FJT의 구현 방법

ω=j인 경우 FJT[8]는 식 (6)과 같은 방법으로 직접 구현하게 되면 N=8인 경우 56개(N(N-1))의 가산과 감산을 필요로 한다[14]. 그러나 FJT에 사용되는 가산과 감산 연산 수는 24개로 감소된다. [14]에 의하면 FJT 행렬계수의 대칭적인 특징과 행렬 분할법을 이용하면 수식연산의 수를 감소시키고 계산과정의 속도를 증가 시킬 수 있다. 입력데이터와 변환데이터의 사이즈가 N인 두개의 벡터를 X와 Y라 하면 출력은 다음과 같다.

$$Y=H_NX=H_N^4H_N^2X=H_N^4T \tag{15}$$

여기서 H_N^4와 H_N^2는 일률적으로 4개와 2개의 0이 아닌 값을 가지고 있는 행렬이다.

$$\begin{bmatrix}Y_1\\Y_2\\Y_3\\Y_4\\Y_5\\Y_6\\Y_7\\Y_8\end{bmatrix}=\begin{bmatrix}1&0&1&0&1&0&1&0\\0&1&0&1&0&1&0&1\\1&0&-1&0&1&0&-1&0\\0&1&0&-1&0&1&0&-1\\1&0&1&0&-1&0&-1&0\\0&1&0&1&0&-1&0&-1\\1&0&-1&0&-1&0&1&0\\0&1&0&-1&0&-1&0&1\end{bmatrix}\begin{bmatrix}1&1&0&0&0&0&0&0\\1&-1&0&0&0&0&0&0\\0&0&1&1&0&0&0&0\\0&0&1&-1&0&0&0&0\\0&0&0&0&1&1&0&0\\0&0&0&0&1&-1&0&0\\0&0&0&0&0&0&1&1\\0&0&0&0&0&0&1&-1\end{bmatrix}\begin{bmatrix}X_1\\X_2\\X_3\\X_4\\X_5\\X_6\\X_7\\X_8\end{bmatrix} \tag{16}$$

$$\begin{bmatrix}Y_1\\Y_2\\Y_3\\Y_4\\Y_5\\Y_6\\Y_7\\Y_8\end{bmatrix}=\begin{bmatrix}1&0&1&0&1&0&1&0\\0&1&0&1&0&1&0&1\\1&0&-1&0&1&0&-1&0\\0&1&0&-1&0&1&0&-1\\1&0&1&0&-1&0&-1&0\\0&1&0&1&0&-1&0&-1\\1&0&-1&0&-1&0&1&0\\0&1&0&-1&0&-1&0&1\end{bmatrix}\begin{bmatrix}T_1\\T_2\\T_3\\T_4\\T_5\\T_6\\T_7\\T_8\end{bmatrix} \tag{17}$$

$$\begin{bmatrix}T_1\\T_2\\T_3\\T_4\\T_5\\T_6\\T_7\\T_8\end{bmatrix}=\begin{bmatrix}X_1+X_2\\X_1-X_2\\X_3+X_4\\X_3-X_4\\X_5+X_6\\X_5-X_6\\X_7+X_8\\X_7-X_8\end{bmatrix} \tag{18}$$

이것은 DA의 원리를 이용하여 수행할 수 있는 FJT 계산의 2단계와 3단계에서 요구되는 가산과 감산 수를 표현한 것이다. 식 (17)은 본래 하다마드 행렬 벡터연산이나 FJT가 하다마드 행렬에 무게를 부과한 것이기 때문에 이식을 이용하여 다음과 같이 표현 할 수 있다.

$$\begin{bmatrix}Y_1\\Y_2\\Y_3\\Y_4\\Y_5\\Y_6\\Y_7\\Y_8\end{bmatrix}=\begin{bmatrix}1&0&1&0&1&0&1&0\\0&1&0&1&0&1&0&1\\1&0&-i&0&i&0&-1&0\\0&1&0&-i&0&i&0&-1\\1&0&i&0&-i&0&-1&0\\0&1&0&i&0&-i&0&-1\\1&0&-1&0&-1&0&1&0\\0&1&0&-1&0&-1&0&1\end{bmatrix}\begin{bmatrix}T_1\\T_2\\T_3\\T_4\\T_5\\T_6\\T_7\\T_8\end{bmatrix} \tag{19}$$

이러한 방법에서 행렬의 형태가 각각의 두 개의 연속되는 행렬이 비슷한 형태의 계수를 가지고 있다. 이것은 ROM을 사용하여 표현할 수 있다. 그림 2는 ROM 테이블에 따라서 ROM과 어큐뮬레이터를 이용하여 구현한 것이다. ROM에 들어가는 내용은 입력벡터에서 주소번지로 발생될 수 있는 모든 경우를 고려하여 ROM에는 각 행에 해당하는 16개의 내용을 미리 저장해 놓은 것이다.

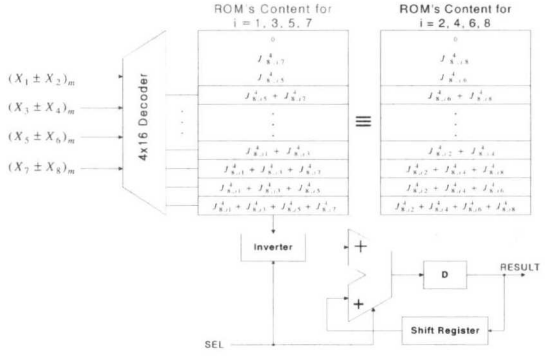


그림 2. 분산 연산 ROM과 누산기 구조

FJT 구조는 그림 3에서 보듯이 변환기로부터 비트 시리얼 형태로 회로에 8개의 입력이 들어간다. 4개의 별도의 RAC 블럭은 다음과 같은 8개의 변환을 계산하고 비트시리얼 형태의 가산과 감산의 나비 구조가 식 (17)의 행렬의 입력 원소를 발생시키는데 사용되어진다.

처음 8-사이클동안 각각의 홀수항이 비트 병렬 형태로 출력된다. 두번째 8-사이클 동안은 각각의 짝수항이 비트 패럴렐 형태로 출력되어진다. 이는 비트 시리얼 가산과 감산의 워드 길이가 n보다 작거나 같을때 입증되어진다. 그렇지 않다면, 연산을 수행하는 동안 8-클럭 지연 외에 또 다른 지연이 있어야 한다. 이것은 부호비트를 포함하여 4비트로 나타내어지는 ROM의 최대값을 가지는 비트 병렬 데이터를 포함한다. 만약 주어지는 가중치가 크다면 비트 수는 최대치를 포함할 수 있도록 비트 수를 늘려주어야 한다. RAC블록 내부의 인버터 신호는 매 n-사이클 마다 ROM의 내용을 2의 보수 형태로 계산하기 위해 사용되어지고, Select 신호는 n-사이클 주기를 가지면서 짝홀수 입력을 선택하기 위해 사용되어진다.

이러한 구조는 분산연산 알고리즘이 사용되어짐으로 인해 곱셈이 가산과 감산으로 표현되어지므로 하드웨어로 쉽게 구현될 뿐만 아니라[17] 수학적으로도 간단하다. 데이터 워드가 n인 이러한 구조에서 복잡도는 $O(2n)$으로 감소한다[14]. 이 그림 3에서 X_1부터 X_8까지의 데이터 입력은 Y_1에서 Y_8값을 출력한다. 각각의 데이터는 이진수 형태로 변화되고 이진 연산이 수행된다. 먼저, 입력 데이터는 바이트 단위로 가산과 감산 연산을 수행하고, 가산연산후의 결과값이 주소에 따라 ROM과 결합하여 최하위 비트부터 최상위 비트까지 이동 누적기(shift accumulator)에서 계산되어 짝수차 Y_{2i}로 출력된다. 이와 비슷한 방식으로, 감산 연산 출력은 8사이클만큼 지연되어 짝수항과 같은 과정을 거쳐, 홀수차 Y_{2i+1}를 출력한다.

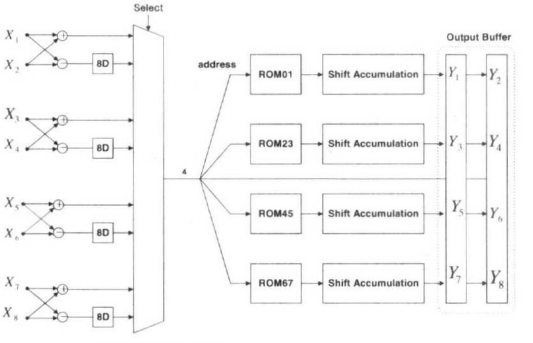


그림 3. 8포인트FJT 구조

V. 결 론

본 논문은 자켓 행렬을 소개하고 이를 비트열로 만들 수 있는 일반식을 제안하였다. 자켓 행렬은 하다마드 행렬 중심부에 무게를 두었으며, 이는 무게와 부호를 결정하는데 사용되어지는 이진 인덱스에 의해 쉽게 표현할 될 수 있다. 웨이팅 계수는 어떤 크기의 행렬일지라도 행과 열의 이진 인덱스 상위 두 비트에 의해 결정되고 무게는 그 두 비트의 Exclusive-OR 연산 결과가 1인 곳의 원소에 부여된다. 무게가 1이면, 자켓 행렬은 하다마드 행렬과 동일하다. 또한, 부호계수는 행과 열의 이진 인덱스 비트들을 AND 연산한 다음 그 결과들을 Exclusive-OR한 합으로 나타낼 수 있다. 자켓 행렬의 역 행렬은 순방향 행렬과 부호는 같고 무게 값만 규칙적으로 변한다. 즉, 순방향과 역방향은 서로 상보적인 관계를 가지므로 하드웨어 구현이 매우 용이하다. 이 구조는 분산연산 알고리즘이 사용되어짐으로 인해 곱셈이 가산과 감산으로 표현되어지므로 하드웨어로 쉽게 구현될 수 있을 뿐만 아니라 수학적으로

도 간단하다. 데이터 워드가 n인 이러한 구조에서 복잡도는 $O(2n)$으로 감소한다.

참 고 문 헌

[1] N.Ahmed and K.R.Rao, *Orthogonal Transforms for Digital Signal Processing*, Berlin, Germany: Springer-Verlag, 1975.

[2] M. H. Lee, "A New Reverse Jacket Transform and Its Fast Algorithm," *IEEE, Trans. On Circuit and System*, vol. 47. no. 1, Jan. 2000.

[3] M. H. Lee, "The complex Reverse Jacket Transform," *22nd Information Theory and Its Application SITA*, Japan Niigata, Nov.31 - Dec.3, 1999.

[4] M. H. Lee, J. Y. Park and S. Y. Hong, "A Simple Binary Index Generation for Reverse Jacket Sequence," *In Proceedings of 2000 International Symposium on Information Theory and Applications (ISITA 2000)*, vol. 1, pp.329-433, Hawaii, USA, Nov. 5-8, 2000.

[5] M. H. Lee and M. Kaveh, "Fast Hadamard transform based on a Simple Matrix Factorization," *IEEE Trans. ASSP-34*, (6), pp.1666~1667, 1986.

[6] M. H. Lee and Y. Yasuda, "Simple Systolic Array for Hadamard Transform," *Electronics Letters*. vol. 26 no.18 pp 1478-1479, 30th August 1990.

[7] M. H. Lee, "A New Reverse Jacket Transform based on Hadamard Matrix," *ISIT2000.2000 IEEE International symposium on Information Theory*, Sorrento (Italy), June 25-30 2000.

[8] M. H. Lee, "Fast Complex Reverse Jacket Transform," Accepted *to IEEE, Trans. on CAS - II.*

[9] M. H. Lee, "The Center Weighted Hadamard Transform," *IEEE, Trans. on Circuit and System*, vol.36. no. 9, pp 1247-1249, Sept.1989.

[10] S. R. Lee and M. H. Lee, "On the Reverse Jacket Matrix for Weighted Hadamard Transform," *IEEE, Trans. on Circuit and System*, vol. 45. no. 3, pp 436-441, Mar. 1998.

[11] M. H. Lee, B. S. Rajan, and J. Y. Park, "A Generalized Reverse Jacket Transform," *IEEE Trans. Circuits Syst. II*, vol.48, no. 7, pp.684-690, July 2001.

[12] H. Jia and M. H. Lee, "Analysis of Relative Difference Set Correlated with Cocyclic Jacket Matrix for Cryptography," *Submit to WISA2001, The 2nd International Workshop on Information Security Application, Sept. 13-14, 2001, Seoul, Korea.*

[13] M. H. Lee, "A New Complex Spreading Jacket QPSK Modulation for WCDMA," *International Workshop on Information Coding Technology*, pp.41-64, July 3-July 4, 2001, Chonju, Korea.

[14] 유경주, 홍선영, 이문호, 정진균, "고속자켓변환의 VLSI 아키텍쳐," 2001 신호처리합동학술대회논문집, vol. 14, no. 1, Sept. 22, 2001.

[15] H. J. and M. H. Lee, "Polynomial Representation of Extending Hadamard Transform," *Proceedings of the 4th MDMC'01*, June 11-12, 2001, Pori, Finland, pp.412-416.

[16] H. Jia., M. H. Lee and J. S. Kim, "Low Density Parity Check Codes with a Code for Iterative Decoding," *Proceedings of the 4th MDMC'01*, June 11-12, 2001, Pori, Finland, pp.59-65.

[17] K. Parhi, *VLSI Digital Signal Processing Systems Design and Implementation*, John Wiley & Sons, Inc. USA, 1999.

[18] S. Rahardja and B. J. Falkowski, "Family of Unified Complex Hadamard Transforms," *IEEE Trans. Circuits and Systems ∏:Analog and Digital Signal Processing. Vol 46. No8*, Aug, 1999.

[19] L. Chang and M. Chang, "A bit level systolic array for Walsh-Hadamard Transforms," *Signal Processing Vol 31, pp341-347*, 1993.

[20] S. Y. KUNG, *VLSI Array Processors*, Prence Hall, USA, 1988.

[21] K.Parhi, *VLSI digital signal processing systems Design and implementation*, John Wiley & Sons, Inc. USA, 1999.

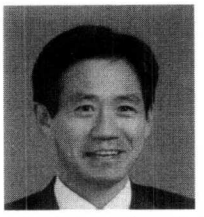
[22] A. Amira, A. Bouridane and P. Milligan, "An FPGA based Walsh Hadamard transforms," *ISCAS 2001. The 2001 IEEE International Symposium on*, Vol 2, pp. 569 -572. 2001.

박 주 용(Ju-yong Park) 정회원

1982년 2월 : 전북대학교 전자공학과 졸업
1986년 2월 : 전북대학교 전자공학과 석사
1994년 2월 : 전북대학교 전자공학과 박사

1991년 3월~현재 : 서남대학교 전기전자멀티미디 학부 부교수
<주관심 분야> 이동통신, 채널코딩, 시퀀스

이 문 호(Moon-ho Lee) 정회원



1990년 2월 : 동경대학 박사
1980년 3월~현재전북대학교 전자 · 정보공학부 교수
1984년~1985년 : 국 미네소타 주립대 전기과 포스트닥터
1983년 : 통신기술사
<주관심 분야> 채널코딩, 이동영상통신, Space-Time 코딩.